REMARKS

Status:

Claims 1 - 17 stand rejected under 35 U.S.C. §103(a) as being unpatentably obvious in view of the teaching of US Pat. No. 6,058,389 to Chandra et al., considered in view of the teaching of US Pat. No. 5,857,180 to Hallmark et al.

Claims 1-17 are presented for reconsideration as explained in the discussion below.

Discussion:

First, considering the recent Office Action at page 2 lines 5-6, it is indicated that "The commit that places the message on the queue is the enqueue operation disclosed by Chandra." Chandra at col. 12, line 12 - col16, line 16 is cited in support of this position. However, referring to Chandra's Fig 6A, it is seen that enqueue is performed as step 608 and Commit occurs later in the logic flow, at step 610. Commit makes the changes made by an enqueue operation permanent and occurs after the enqueue operation has finished. Indeed, at Chandra col. 14, lines 4-6, it is indicated, "In step 610 the process calls **another kernel function** to commit the recursive transaction" (bolding added). Again, note step 610 is a separate step subsequent to step 608 in the logic flow. The logic branch including step 604 does not show a commit step and the calling program apparently prevents any rollback (Chandra, col. 13., lines 38-44).

Applicant's claims call for "assigning an index key to a message **in response to commit** of the operation of putting the message on the queue." In Chandra's logic of Fig. 6A, that would be an operation to assign an index key in the flow after the COMMIT step 610 (and earlier ENQUEUE step 608), an operation Chandra does not teach.

For the flow branch of step 604, note at col 13, lines 41-43, the Chandra specification indicates "changes to the queue are not made until the calling application program indicates that the transaction should commit." At col. 11, lines 49-54, it is indicated that: *"An application may specify that a specific request is a transaction itself, thereby making its result immediately available. For example, messages can be made available to other applications either immediately after an enqueue or dequeue operation is completed or only after the transaction is completed."* Chandra does not teach an assignment of an index key in response to commit of the operation of putting the

- 8 -

message on the queue in that event either. Note, Applicant's claims (see especially claims 1 and 13) now emphasize that commit occurs subsequent to the putting of the message.

Further, at page 2 of the recent Office Action, lines 13-15 it is indicated that "Chandra teaches the timing of assignment, assigning the index key in response to commit/enqueue." And that "There is no lock involved in the enqueue operation."

As discussed above, enqueue and commit are separate operations. Commit follows and makes the change permanent (again see Chandra's Fig. 6A). Merging the two operations with a "/" does overcome the lack of a teaching of logic responding to the commit (step 610) that assigns an index key. Again, there is no such logic after step 610 and merging steps 608 and 610 does not create one.

As regards the lock, see Chandra at col. 31, lines 39-47 which indicates *a commit must occur before the operation is carried out.* That sounds like a lock requirement. There is no deferred index key shown in Chandra Fig. 6A to prevent the operation. Also in this regard, see Attachment A, The IBM Dictionary of Computing (several pages, including p. 119) regarding the definition of "COMMIT". Item 4 of the definition refers to SQL(which Chandra recommends at the bottom of col. 12) and indicates *"When a commit operation occurs, the locks are released to allow other applications to use the changed data."*

The lock logic prevents access before commit; and, as Applicant has taught and claimed, this result is advantageously achieved without lock logic by deferring assignment of the index key to occur after commit (Applicant's spec. at p. 9, lines 7-17). By so keeping the index key unavailable, Applicant prevents premature access without lock logic and testing of a lock state before accessing.

Again Chandra does not teach a deferred assignment of an index key. And, the Hallmark teaching (US Pat. 5,597,180) does not overcome the deficiency. At col. 14, lines 3-6 it is indicated *a scheduling system* keeps track of the availability of data. That is not a deferred assignment of an index key, either.

It is believed that the claims (e.g. claim1 at line 3 and claim 13 at line 6) emphasize the assigning of the index key **in response to commit** (the **subsequent** commit of a put for placing the message on the queue). This is the claimed advantageous timing that

- 9 -

Applicant has recognized serves to reduce processing overhead relative to the prior art, as explained above.

In accordance with the foregoing, it is believed the claims now clearly identify Applicants advantageous commit-responsive assignment of an index key - a deferred assignment that avoids the need for special logic to prevent premature access of shared data. Hence applicants earnestly solicit early notice that this case has been placed in condition for allowance.

Respectfully Submitted,

George E. Grosser

Reg. No. 25,629

c/o

IBM Corp.

Dept. T81/Bldg. 503 PO Box 12195

Research Triangle Park. NC 27709

(919)968-7847

Fax 919-254-4330

EMAIL: gegch@prodigy.net

Attachment A *p.1*

IBM DICTIONARY OF COMPUTING

Compiled and edited by
GEORGE McDANIEL

McGRAW-HILL, INC.
New York San Francisco Washington, D.C. Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

Attachment A p. 2

Limitation of Liability

While the Editor and Publisher of this book have made reasonable efforts to ensure the accuracy and timeliness of the information contained herein, neither the Editor nor the Publisher shall have any liability with respect to loss or damage caused or alleged to be caused by reliance on any information contained herein.

67890 DOC/DOC 9987

ISBN 0-07-031488-8 (HC)
ISBN 0-07-031489-6 (PBK)

The sponsoring editor for this book was Daniel A. Gonneau and the production supervisor was Thomas G. Kowalczyk.

Printed and bound by R. R. Donnelley & Sons Company.

Tenth Edition (August 1993)

This is a major revision of the *IBM Dictionary of Computing*, SC20-1699-8, which is made obsolete by this edition. Changes are made periodically to the information provided herein.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Comments may be addressed to IBM Corporation, Department E37/656, P. O. Box 12195, Research Triangle Park, NC 27709.

International Edition

Copyright © 1994 by International Business Machines Corporation. Exclusive rights by McGraw-Hill, Inc. for manufacture and export. This book cannot be re-exported from the country to which it is consigned by McGraw-Hill. The International Edition is not available in North America.
When ordering this title, use ISBN 0-07-113383-6.

This book is printed on acid-free paper.

Attachment A p-3

commit

mand profile

jort Program Product that processes control commands and passes procedure commands and operation tol language statements to the initiator. (3) In NetView program, a module designed to perform a ific function for the user. Users can write mand processors in assembler language or in a .level language (HLL); command processors are oked as commands.

mand profile See profile.

mand programming language A language that ws programming by the use of commands rather r by writing statements in a conventional programing language.

mand prompt A displayed character or string of aracters that indicates that a user may enter a mand to be processed.

mand/response application In the Network ater Interconnect Manager and Agent programs, a iction that allows a NetView operator to execute mands in a carrier management system.

mand retry A channel and control unit procedure at causes a command to be retried without requiring I/O interruption.

mand scan In CMS, a routine that scans the mand line entered and converts it to a standard MS parameter list.

mand statement A job control statement that is ed to issue commands to the system through the put stream.

mand string In AS/400 query management, a haracter string that contains a query command.

mand substitution In the AIX operating system, e ability to capture the output of any command as a alue to another command by placing that command ine within (grave accents). The shell first runs the command or commands enclosed within the grave cents and then replaces the whole expression, including grave accents, with their output. This feature is often used in assignment statements.

mand virtual terminal In the AIX operating system, the virtual terminal that becomes active when the command window hot key is pressed. See also command window hot key.

mand window hot key In the AIX operating system, a key combination that activates the command virtual terminal. The command window hot key combination is Alt-Action on the keyboard, the two buttons on a mouse, or button number 4 on a tablet. See also command virtual terminal.

command word In the AIX operating system, the name of the 16-bit units used for storing graphic primitive strings. The first command word determines the itive strings. The first command word determines the primitive type and sets the length of the string. Subsequent command words contain information in multiples of quid, or 4 bits of data.

comment (1) In programming languages, a language construct for the inclusion of text in a program and having no impact on the execution of the program. Comments are used to explain certain aspects of the program. (1) (2) A statement used to document a program or file. Comments include information that may be helpful in running a job or reviewing an output listing. (3) In the C language, a token that consists of one or more lines, delimited by /* and */. Comments can be written anywhere in the program. (4) In Pascal, a token consisting of characters on one or more lines, delimited by /* */, (* *), or [ ] . Comments can be written anywhere in the program. (5) In SQL, source program information that is not translated by the compiler. The format of a comment is language specific. (6) Synonymous with computer program annotation, note, remark.

comment-entry In COBOL, an entry in the Identification Division that may be any combination of characters from the character set of a computer.

comment line In COBOL, a source program line represented by an asterisk (*) in the indicator area of the line and any characters from the character set of the computer in area A and area B of that line.

comment statement A source language statement that has no effect other than to be reproduced on an output listing.

commercial instruction set A combination of instructions of the standard instruction set and the decimal feature.

Commission Internationale de l'Éclairage (CIE) An international committee that develops color standards.

commit (1) In DPPX/DTMS, to ensure action on requests made by a program during its current scope of recovery to change resettable or recreatable databases and to execute transactions. (2) To end the current scope of recovery and begin a new one. (3) To make all changes permanent that were made to one or more database files since the last commit or rollback operation, and make the changed records available to other users. (4) In SQL, the process that allows data changed by one application or user to be used by other applications or users. When a commit operation occurs, the locks are released to allow other applications to use the changed data. (5) A service that performs commit actions. (6) In IMS/VS, an

Attachment A p. 4

indication in an application program that a section of work is done and that the data it has modified or created is consistent and complete.

commit cycle In System/38, the sequence of changes made between commitment boundaries

commit cycle identifier In System/38, the journal sequence number associated with the start commitment entry that is used to identify the journal entries in a particular commit cycle.

commit identifier In the AS/400 system and System/38, the information that associates a commitment operation with a specific set of database changes. The commit ID is placed in the notify object if a system or routing step failure occurs, or if uncommitted changes exist when a routing step ends normally. See also notify object.

commitment, concurrency, and recovery (CCR) An application service element that controls operations performed by two or more application processes on shared data to ensure that the operations are performed either completely or not at all. (T)

commitment boundary In a commitment controlled environment, a point at which there are no changes to a database file pending within a job.

commitment control In FORTRAN, a means of grouping file operations that allows the processing of a group of database changes as a single unit or the removal of a group of database changes as a single unit.

commitment definition In the AS/400 system, information used by the system to maintain the commitment control environment throughout a routing step and, in the case of a system failure, throughout an initial program load IPL. This information is obtained from the Start Commitment Control command, which establishes the commitment control environment, and the file open information in a routing step.

commitment function In CICS/VS, a synchronization processing function that allows a transaction program involved in a synchronized unit of work to ensure that all restorable resources associated with that unit are brought to a constant state of update. See also preparation function.

commit operation An operation that saves a file in permanent storage.

commit point In SQL, the point in time when data is considered to be consistent.

committed change A database change that will not be backed out during system failure. Changes made by a

logical unit of work are committed when the synchronization point at the end of the logical unit of work is complete.

committed state The state of the resources associated with a synchronized unit of work following successful execution of the commitment function.

common action In SAA Common User Access architecture, one of a set of actions that has common meaning across all applications; for example, exit, cancel, help.

common address space section (CASS) In DPPX, subpools in all address spaces that are associated with the same real storage location. This area is addressable by any process running in any address space. It is made up of the read-only and read-write subpools.

common area (1) A control section used to reserve a main storage area that can be referred to by other modules. (2) In OS/VS2, the area of virtual storage that is addressable by all address spaces. (3) In FORTRAN, a storage area that is used for communication between a main program and one or more subprograms.

common battery central office A central office that supplies transmitter and signal current for its associated stations and for signaling by the central office equipment from a power source located in the central office. See also tip.

common block In XL FORTRAN, a storage area that can be referred to by a calling program and one or more subprograms.

common buffer In DPCX, a block of processor storage made up of four 256-byte blocks that can be used by all active programs.

common carrier See communication common carrier.

common communication adapter (CCA) A general purpose adapter that inserts or removes control information and converts message data into an appropriate form for the terminal in which the data are used.

Common Communications Support Protocols and conventions for connecting systems and software. One of the three SAA architectural areas. See also Common Programming Interface, Common User Access architecture.

common field A field that can be accessed by two or more independent routines. (A)

common key In COBOL, the key fields that are common to all record formats in the file starting with